


Brief Announcement: The Expressive Power of Uniform Population Protocols with Logarithmic Space

Philipp Czerner   

Technical University of Munich, Germany

Vincent Fischer  

Technical University of Munich, Germany

Roland Guttenberg  

Technical University of Munich, Germany

Abstract

Population protocols are a model of computation in which indistinguishable mobile agents interact in pairs to decide a property of their initial configuration. Originally introduced by Angluin et. al. in 2004 with a constant number of states, research nowadays focuses on protocols where the space usage depends on the number of agents. The expressive power of population protocols has so far however only been determined for protocols using $o(\log n)$ states, which compute only semilinear predicates, and for $\Omega(n)$ states. This leaves a significant gap, particularly concerning protocols with $\Theta(\log n)$ or $\Theta(\text{polylog } n)$ states, which are the most common constructions in the literature. In this paper we close the gap and prove that for any $\varepsilon > 0$ and $f \in \Omega(\log n) \cap \mathcal{O}(n^{1-\varepsilon})$, both uniform and non-uniform population protocols with $\Theta(f(n))$ states can decide exactly $\text{NSPACE}(f(n) \log n)$.

2012 ACM Subject Classification Theory of computation \rightarrow Distributed computing models

Keywords and phrases Population Protocols, Uniform, Expressive Power

Digital Object Identifier 10.4230/LIPIcs.DISC.2024.44

Related Version *Full Version*: <https://arxiv.org/abs/2408.10027> [15]

1 Introduction

Population protocols are a model of computation in which indistinguishable mobile agents randomly interact in pairs to decide whether their initial configuration satisfies a given property. The decision is taken by *stable consensus*; eventually all agents agree on whether the property holds or not, and never change their mind again. While originally introduced to model sensor networks [4], population protocols are also very close to chemical reaction networks [23], a model in which agents are molecules and interactions are chemical reactions.

Originally agents were assumed to have a finite number of states [4, 5, 6], however many predicates then provably require at least $\Omega(n)$ time to decide [21, 7, 1], as opposed to recent breakthroughs of $\mathcal{O}(\log n)$ time using $\mathcal{O}(\log n)$ number of states (in some cases even $\mathcal{O}(\log \log n)$ states) for important tasks like leader election [9] and majority [19]. Limiting the number of states to logarithmic is important in most applications, especially the chemical reaction setting, since a linear in n number of states would imply the unrealistic number of approximately 10^{23} different chemical species. Therefore most recent literature focusses on the polylogarithmic time and space setting, and determines time-space tradeoffs for various important tasks like majority [3, 1, 2, 22, 8, 19], leader election [1, 22, 9] or estimating/counting the population size [20, 16, 10, 17, 18].



© Philipp Czerner, Vincent Fischer, and Roland Guttenberg;
licensed under Creative Commons License CC-BY 4.0

38th International Symposium on Distributed Computing (DISC 2024).

Editor: Dan Alistarh; Article No. 44; pp. 44:1–44:7



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

This leads to the interesting open problem of characterizing the class of predicates which can be computed in polylogarithmic time using a logarithmic or polylogarithmic number of states. There is however a fundamental problem with working on this question: Despite the focus on $\mathcal{O}(\log n)$ number of states in recent times, the expressive power for this number of states (regardless of time) is still unknown.

More precisely, there is a gap in the existing literature: protocols with $f(n) \in \Omega(n)$ states are known to have expressive power $\text{NSPACE}(n \log f(n))$ [14], i.e. symmetric predicates in $\text{NSPACE}(n \log f(n))$, while a subclass of protocols with $o(\log n)$ states can only compute semilinear predicates [6, 14]. The latter result applies only to *uniform* population protocols, i.e. protocols where the transitions are independent of the size of the population.

However, many constructions in the literature have e.g. $\Theta(\log n)$ or $\Theta(\text{polylog } n)$ states. This important case is not covered by the existing results. To the best of our knowledge, the only research in this direction is [12], where the expressive power is characterised for $\text{polylog}(n)$ number of states for a similar model – not population protocols themselves. Most importantly, their results do not lead to a complete characterization for $\Theta(\log n)$ states since they lose some log factors in their characterization of $\text{polylog}(n)$.

In this paper, we fill the gap by proving that for functions $f(n) \in \Omega(\log n) \cap \mathcal{O}(n^{1-\varepsilon})$, where $\varepsilon > 0$, population protocol with $f(n)$ states compute exactly $\text{NSPACE}(f(n) \cdot \log n)$, i.e. the symmetric predicates computable by a non-deterministic Turing machine using $\mathcal{O}(f(n) \cdot \log n)$ space. This result applies to both uniform and non-uniform protocols. (The function f needs to fulfil some technical conditions.)

With this result, the expressive power of uniform population protocols is characterised in all cases, and for non-uniform protocols it is characterised in the case of $\Omega(\log n)$ states. (A slight gap between $\mathcal{O}(n^{1-\varepsilon})$ and $\Omega(n)$ remains.)

2 Preliminaries

► **Definition 1.** A protocol scheme \mathcal{P} is a 5-tuple $(Q, \Sigma, \delta, I, O)$ of

- a (not necessarily finite) set of states Q ,
- a finite input alphabet Σ ,
- a (partial) transition function $\delta : Q \times Q \rightarrow Q \times Q$,
- an injective input mapping $I : \Sigma \rightarrow Q$,
- an output mapping $O : Q \rightarrow \{0, 1\}$.

A *configuration* of \mathcal{P} is a finite multiset $C \in \mathbb{N}^Q$, which represents a collection of agents with states in Q . A step $C \rightarrow C'$ in \mathcal{P} occurs by choosing two agents from C and letting them interact via δ , i.e. if their states are p, q in C , then their new states in C' will be $\delta(p, q)$.

We write \rightarrow^* for the reflexive and transitive closure of \rightarrow , and say that a configuration C' is *reachable* from C if $C \rightarrow^* C'$. The input to \mathcal{P} consists of a multiset $w \in \mathbb{N}^\Sigma$. Every input w can be mapped to its corresponding *initial* configuration by applying I to every letter in w .

A configuration C is a *b-consensus* for $b \in \{0, 1\}$ if $O(q) = b$ for all q such that $C(q) \neq 0$, i.e. if every state which occurs in the configuration has output b . A configuration C is *stable with output b* if every configuration C' reachable from C is a *b-consensus*.

A *run* ρ is an infinite sequence of configurations $\rho = (C_0, C_1, \dots)$ such that $C_i \rightarrow C_{i+1}$ for all $i \in \mathbb{N}$. A run is *fair* if for all configurations C which occur infinitely often in ρ , i.e. such that there are infinitely many i with $C_i = C$, every configuration C' reachable from C occurs infinitely often in ρ . A run has *output b* if some configuration C_i along the run is stable with output b (and hence all C_j for $j \geq i$ are also stable with output b).

An input $w \in \mathbb{N}^\Sigma$ has *output* b if every fair run starting at its corresponding initial configuration $\hat{I}(w)$ has output b . The protocol scheme \mathcal{P} *computes* a predicate $\varphi: \mathbb{N}^\Sigma \rightarrow \{0, 1\}$ if every input w has output $\varphi(w)$.

Let us provide an example which also shows how to treat infinite sets Q .

► **Example 2.** Consider $Q := \{0\} \cup \{2^i \mid i \in \mathbb{N}\}$, and define $\delta(2^i, 2^i) = (2^{i+1}, 0)$. Let $\Sigma = \{x\}$, and let $x \mapsto 2^0$ be the input mapping. Then a configuration is initial if every agent is in state 2^0 . Intuitively this protocol will eventually end up with the binary representation of the number of agents. Namely each transition preserves the total sum of all agents' values, and every transition increases the number of agents in 0, so this protocol in fact always reaches a terminal configuration.

Regarding the infinite state space, intuitively the protocol uses $\lfloor \log n \rfloor + 2$ states, namely $\lfloor \log n \rfloor + 1$ powers of two and 0. The other states cannot be reached with n agents.

Accordingly we now define the state complexity of a protocol scheme. A state $q \in Q$ is *coverable* from some initial configuration C_0 if there exists a configuration C reachable from C_0 which fulfils $C(q) > 0$. The *state complexity* $S(n)$ of \mathcal{P} for n agents is the number of states $q \in Q$ which are coverable from some initial configuration with n agents.

► **Example 3.** In the scheme of Example 2, let C_n be the unique initial configuration with n agents, i.e. $C_n(2^0) = n$ and $C_n(q) = 0$ otherwise. For $n \geq 2$, the states coverable from C_n are exactly $\{0\} \cup \{2^i \mid i \leq \log n\}$. Hence the state complexity is $S(n) = \lfloor \log n \rfloor + 2$.

As defined so far, protocol schemes are not necessarily computable. Hence actual population protocols require some uniformity condition.

► **Definition 4.** A uniform population protocol $\mathcal{P} = (Q, \Sigma, \delta, I, O)$ is a protocol scheme together with a bijection $Q \rightarrow \{0, 1\}^*$ to represent Q via binary strings, such that the functions δ, I, O are computable by linear space Turing-machines (TMs).

We remark that “linear space” then in terms of our n , the number of agents, is $\mathcal{O}(\log S(n))$ space (since the input of the machine is a representation of a state).

In the literature on uniform population protocols, e.g. [13, 14, 20, 16], often agents are defined as TMs and states hence automatically assumed to be represented as binary strings. We avoid talking about the exact implementation of a protocol via TMs because it introduces an additional logarithm in the number of states and potentially confuses the reader, while most examples are clearly computable.

► **Example 5.** In the protocol scheme of Example 2 we represent states by the binary representation of the exponent. Clearly incrementing natural numbers or setting the number to a fixed value are possible by a linear space TM, hence this is a uniform population protocol.

Next we define a more general class of population protocols, which we call weakly uniform. This class includes all known population protocols, and our results also hold for this class, which shows that having a different protocol for every n does not strengthen the model.

► **Definition 6.** A finite population protocol is a protocol scheme with a finite set Q .

A population protocol \mathcal{P} is an infinite family $(\mathcal{P}_n)_{n \in \mathbb{N}} = (Q_n, \Sigma, \delta_n, I_n, O_n)_n$ of finite population protocols. The state complexity for inputs of size n is $S(n) := |Q_n|$.

\mathcal{P} is weakly uniform if there exist TMs M_δ, M_I, M_O using $\mathcal{O}(S(n))$ space which compute δ_n, I_n and O_n , respectively, taking n as additional input.

The configurations of \mathcal{P} with n agents are exactly the configurations of \mathcal{P}_n with n agents, and accordingly the semantics of steps, runs and acceptance are inherited from \mathcal{P}_n .

The protocol for a given population size n is allowed to differ completely from the protocol for $n - 1$ agents, as long as TMs are still able to evaluate transitions, input and output. Usually this is not fully utilised, with the most common case of a non-uniform protocol being that $\log n$ is encoded into the transition function [19].

Clearly uniform population protocols are weakly uniform. Namely let $\mathcal{P} = (Q, \Sigma, \delta, I, O)$ be a protocol scheme. Then for every $n \in \mathbb{N}$ we let Q_n be the set of states coverable by some initial configuration with n agents, similar to the definition of state complexity, and define $\mathcal{P}_n := (Q_n, \Sigma, \delta_n|_{Q_n^2}, I, O|_{Q_n})$, where $f|_A$ is the restriction of f to inputs in A . This protocol family computes the same predicate, and is weakly-uniform with the same state complexity.

Next we define the complexity classes for our main result. Let $f: \mathbb{N} \rightarrow \mathbb{N}$ be a function. f is space-constructible if there exists a TM M which computes f using $\mathcal{O}(f(n))$ space. Given a space-constructible function $f: \mathbb{N} \rightarrow \mathbb{N}$, we denote by $\text{NSPACE}(f(n))$ the class of predicates computable by a non-deterministic Turing-machine in $\mathcal{O}(f(n))$ space, and by $\text{SNSPACE}(f(n))$ the class of symmetric (i.e. only depending on the count of letters) predicates in $\text{NSPACE}(f(n))$. Similarly, let $\text{UPP}(f(n))$ be the class of predicates computable by uniform population protocols with $\mathcal{O}(f(n))$ space, and $\text{WUPP}(f(n))$ be the class of predicates computable by weakly-uniform population protocols with $\mathcal{O}(f(n))$ space.

3 Main Result

We give a characterisation for the expressive power of both uniform and weakly uniform population protocols with $f(n)$ states, where $f \in \Omega(\log n) \cap \mathcal{O}(n^{1-\varepsilon})$, for some $\varepsilon > 0$. For technical reasons, we must place a few limitations on $f(n)$ (see the full paper [15] for details). We will refer to a function f fulfilling these requirements as *reasonable*.

Our bound applies to uniform and weakly uniform protocols. As mentioned in the previous section, the latter includes, to the best of our knowledge, all non-uniform constructions from the literature.

► **Theorem 7.** *Let $\varepsilon > 0$ and let $f \in \Omega(\log n) \cap \mathcal{O}(n^{1-\varepsilon})$ be reasonable. Then*

$$\text{UPP}(f(n)) = \text{WUPP}(f(n)) = \text{SNSPACE}(f(n) \cdot \log n)$$

Proof. This will follow from Proposition 8 and Theorem 9. ◀

In particular, we have $\text{UPP}(\log n) = \text{WUPP}(\log n) = \text{SNSPACE}(\log^2 n)$.

► **Proposition 8 (Upper Bound).** *Let $\varepsilon > 0$ and let $f \in \Omega(\log n) \cap \mathcal{O}(n^{1-\varepsilon})$ be space-constructible. Then*

$$\text{UPP}(f(n)) \subseteq \text{WUPP}(f(n)) \subseteq \text{SNSPACE}(f(n) \log n)$$

Proof (sketch). $\text{UPP}(f(n)) \subseteq \text{WUPP}(f(n))$ is trivial/was explained in Section 2. $\text{WUPP}(f(n)) \subseteq \text{SNSPACE}(f(n) \log n)$ can be shown using a reduction to a reachability problem in the configuration graph as in [11]. ◀

Our main contribution is the proof of the lower bound:

► **Theorem 9 (Lower Bound).** *Let $\varepsilon > 0$ and let $f \in \Omega(\log n) \cap \mathcal{O}(n^{1-\varepsilon})$ be reasonable. Then*

$$\text{SNSPACE}(f(n) \log n) \subseteq \text{UPP}(f(n))$$

Proof (sketch). We construct a uniform population protocol $\mathcal{P} = (Q, \Sigma, \delta, I, O)$, simulating a counter machine with $|\Sigma|$ input counters using space $\mathcal{O}(2^{f(n)\log n})$, which is equivalent to a $\mathcal{O}(f(n)\log n)$ space-bounded Turing machine.

Initialisation. Our first goal is to reach a configuration where the number of agents n is known. By this we mean, that we want to have $\lceil \log n \rceil + 1$ uniquely identifiable “counter agents”, each of which stores one bit of the binary representation of n . We use a similar approach as in Example 2 to achieve this:

$$\begin{aligned} (\text{Ctr}, i, 1), (\text{Ctr}, i, 1) &\mapsto (\text{Ctr}, i + 1, 1), (\text{Ctr}, i, 0) && \text{for } i \in \mathbb{N} && \langle \text{counter} \rangle \\ (\text{Ctr}, i, 0), (\text{Ctr}, i, b) &\mapsto (\text{Ctr}, i, b), (\text{Ldr}, i + 1) && \text{for } i \in \mathbb{N} \end{aligned}$$

Here $(\text{Ctr}, i, 1)$ encodes that the i -th bit in the binary representation of n is set, while $(\text{Ctr}, i, 0)$ represents an unset bit. The first transition is analogous to Example 2, but instead of simply sending the second agent to state 0, it also remembers which bit it represents. The second transition gets rid of additional agents storing the same bit.

Among the remaining agents we now want to elect one leader, who knows how many bits n has. We will refer to all agents which are neither counters nor a leader as *free*:

$$\begin{aligned} (\text{Ldr}, i), (\text{Ldr}, j) &\mapsto (\text{Ldr}, j), \text{Free} && \text{for } i, j \in \mathbb{N}, i \leq j && \langle \text{leader} \rangle \\ (\text{Ldr}, i), (\text{Ctr}, j, b) &\mapsto (\text{Ldr}, j), (\text{Ctr}, j, b) && \text{for } i, j \in \mathbb{N}, i \leq j \end{aligned}$$

The first transition here is a standard leader election, the second informs the leader of the number of bits required to store n .

At some point a configuration will be reached, where the binary counting and leader elections have been completed. Note that there is no way of telling for certain when this is the case. For now, we will assume that we have reached such a configuration and describe how to solve this problem later on.

Once n is known, we gain the ability to loop over all agents: each of the counter agents stores an additional, initially unset, bit. Every agent stores a marker flag. The Leader can then apply operations to all agents by sequentially interacting with them and setting the marker flag. Each time the leader interacts with an agent which has the marker flag unset, it increments the second value stored in the counter agents. If at some point both values match, then, as the first value is n , all of the agents must have been marked. In particular this allows the leader to check if an agent with a certain state does not exist. Normally this is quite difficult for population protocols, as agents in the queried state might not take part in any interactions for an arbitrarily long time.

Simulating Counter Machines. Often, when population protocols need to simulate some type of counter, either a unary [5], or binary encoding [12], is used. Neither approach works for us, as we need to be able to count up to $2^{f(n)\log n}$, but a unary encoding with n agents is bounded by n , and a binary encoding with $f(n)$ distinguishable digits is bounded by $2^{f(n)}$. Instead we use a mixed-radix positional encoding with the base $b_i \in \Omega(\frac{n}{f(n)})$ for every digit i . To achieve this, the leader evenly divides the remaining free agents into $\Omega(f(n))$ groups, each encoding one digit. Recall that the leader can detect when no free agents remain, so it will know when this process is finished. Within each digit unary counting is used, that is, each agent in that digit stores one counter bit and the overall value of the digit is the sum of all counter bits. The commands of the counter machine involve manipulating these digits, by either incrementing, or decrementing the encoded values, as well as checking whether they are zero. For the latter, the leader again uses the ability to detect whether a state is present in the population.

Resets. The counter machine simulation described in the previous section relies on the looping and absence checks enabled by the data structures set up during initialisation. However, there is no way of being certain that the initialisation has finished. We solve this by raising a dirty flag each time a transition from the initialisation phase occurs. When seen by the leader, this will trigger a reset, where the leader will move all agents back to state *Free*, once again relying on being able to count the number of agents. When the last reset occurs, the counter agents must encode the correct value of n , and the leader is thus able to iterate over all agents. Care must be taken s.t. other agents do not interact with agents in *Free* while the reset is ongoing, e.g. when only half of the agents have moved to *Free*, and the others are still some intermediate states. ◀

References

- 1 Dan Alistarh, James Aspnes, David Eisenstat, Rati Gelashvili, and Ronald L. Rivest. Time-space trade-offs in population protocols. In *SODA 2017*, pages 2560–2579. SIAM, 2017. doi:10.1137/1.9781611974782.169.
- 2 Dan Alistarh and Rati Gelashvili. Recent algorithmic advances in population protocols. *SIGACT News*, 49(3):63–73, 2018. doi:10.1145/3289137.3289150.
- 3 Dan Alistarh, Rati Gelashvili, and Milan Vojnovic. Fast and exact majority in population protocols. In *PODC*, pages 47–56. ACM, 2015. doi:10.1145/2767386.2767429.
- 4 Dana Angluin, James Aspnes, Zoë Diamadi, Michael J. Fischer, and René Peralta. Computation in networks of passively mobile finite-state sensors. In *PODC 2004*, pages 290–299. ACM, 2004. doi:10.1145/1011767.1011810.
- 5 Dana Angluin, James Aspnes, and David Eisenstat. Fast computation by population protocols with a leader. In *DISC*, volume 4167 of *Lecture Notes in Computer Science*, pages 61–75. Springer, 2006. doi:10.1007/11864219_5.
- 6 Dana Angluin, James Aspnes, David Eisenstat, and Eric Ruppert. The computational power of population protocols. *Distributed Comput.*, 20(4):279–304, 2007. doi:10.1007/S00446-007-0040-2.
- 7 Amanda Belleville, David Doty, and David Soloveichik. Hardness of computing and approximating predicates and functions with leaderless population protocols. In *ICALP*, volume 80 of *LIPICs*, pages 141:1–141:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPICs.ICALP.2017.141.
- 8 Petra Berenbrink, Robert Elsässer, Tom Friedetzky, Dominik Kaaser, Peter Kling, and Tomasz Radzik. Time-space trade-offs in population protocols for the majority problem. *Distributed Comput.*, 34(2):91–111, 2021. doi:10.1007/S00446-020-00385-0.
- 9 Petra Berenbrink, George Giakkoupis, and Peter Kling. Optimal time and space leader election in population protocols. In *STOC*, pages 119–129. ACM, 2020. doi:10.1145/3357713.3384312.
- 10 Petra Berenbrink, Dominik Kaaser, and Tomasz Radzik. On counting the population size. In *PODC*, pages 43–52. ACM, 2019. doi:10.1145/3293611.3331631.
- 11 Michael Blondin, Javier Esparza, and Stefan Jaax. Expressive power of broadcast consensus protocols. In *CONCUR*, volume 140 of *LIPICs*, pages 31:1–31:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICs.CONCUR.2019.31.
- 12 Olivier Bournez, Johanne Cohen, and Mikaël Rabie. Homonym population protocols. *Theory Comput. Syst.*, 62(5):1318–1346, 2018. doi:10.1007/S00224-017-9833-2.
- 13 Ioannis Chatzigiannakis, Othon Michail, Stavros Nikolaou, Andreas Pavlogiannis, and Paul G. Spirakis. Passively mobile communicating logarithmic space machines. *CoRR*, abs/1004.3395, 2010. arXiv:1004.3395.
- 14 Ioannis Chatzigiannakis, Othon Michail, Stavros Nikolaou, Andreas Pavlogiannis, and Paul G. Spirakis. Passively mobile communicating machines that use restricted space. *Theor. Comput. Sci.*, 412(46):6469–6483, 2011. doi:10.1016/J.TCS.2011.07.001.

- 15 Philipp Czerner, Vincent Fischer, and Roland Guttenberg. The expressive power of uniform population protocols with logarithmic space, 2024. [arXiv:2408.10027](https://arxiv.org/abs/2408.10027).
- 16 David Doty and Mahsa Eftekhari. Efficient size estimation and impossibility of termination in uniform dense population protocols. In *PODC*, pages 34–42. ACM, 2019. doi:10.1145/3293611.3331627.
- 17 David Doty and Mahsa Eftekhari. A survey of size counting in population protocols. *Theor. Comput. Sci.*, 894:91–102, 2021. doi:10.1016/J.TCS.2021.08.038.
- 18 David Doty and Mahsa Eftekhari. Dynamic size counting in population protocols. In *SAND*, volume 221 of *LIPICs*, pages 13:1–13:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICs.SAND.2022.13.
- 19 David Doty, Mahsa Eftekhari, Leszek Gasieniec, Eric E. Severson, Przemyslaw Uznanski, and Grzegorz Stachowiak. A time and space optimal stable population protocol solving exact majority. In *FOCS*, pages 1044–1055. IEEE, 2021. doi:10.1109/FOCS52979.2021.00104.
- 20 David Doty, Mahsa Eftekhari, Othon Michail, Paul G. Spirakis, and Michail Theofilatos. Brief announcement: Exact size counting in uniform population protocols in nearly logarithmic time. In *DISC*, volume 121 of *LIPICs*, pages 46:1–46:3. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPICs.DISC.2018.46.
- 21 David Doty and David Soloveichik. Stable leader election in population protocols requires linear time. In *DISC*, volume 9363 of *Lecture Notes in Computer Science*, pages 602–616. Springer, 2015. doi:10.1007/978-3-662-48653-5_40.
- 22 Robert Elsässer and Tomasz Radzik. Recent results in population protocols for exact majority and leader election. *Bull. EATCS*, 126, 2018. URL: <http://bulletin.eatcs.org/index.php/beatcs/article/view/549/546>.
- 23 David Soloveichik, Matthew Cook, Erik Winfree, and Jehoshua Bruck. Computation with finite stochastic chemical reaction networks. *Nat. Comput.*, 7(4):615–633, 2008. doi:10.1007/S11047-008-9067-Y.